Τεχνητή Νοημοσύνη
Εαρινό Εξάμηνο 2025
Διδάσκων: Α. Λύκας

- Ομάδες **δύο ή τριών** (κατά προτίμηση) **φοιτητών**.
- Φοιτητές που έχουν βαθμολογηθεί σε προηγούμενα έτη δεν δικαιούνται επανεξέτασης.
- Γλώσσα προγραμματισμού: **C ή Java** (όχι Python)
- Δεκτές για εξέταση γίνονται μόνο ασκήσεις που είναι ολοκληρωμένες, δηλ. τα προγράμματα μεταγλωττίζονται και εκτελούνται στους υπολογιστές του Τμήματος π.χ. opti3060ws03. Μην υποβάλετε κώδικα Java που απαιτεί το περιβάλλον eclipse για να μεταγλωττιστεί και να εκτελεστεί.
- **Δήλωση ομάδων**: e-mail στον διδάσκοντα με (ΑΜ, ονοματεπώνυμο) μελών της ομάδας **μέχρι 11 Απριλίου 2025** (αυστηρή προθεσμία).
- Η δήλωση συνεπάγεται υποχρέωση υποβολής των εργασιών.
- **Προθεσμία υποβολής εργασιών: 11 Μαΐου 2025** (αυστηρή προθεσμία).
- Η **υποβολή** θα γίνει με χρήση της εντολής **turnin** ως εξής:

  turnin assignment@myy602 <your_filename>
- Θα δημιουργήσετε δύο καταλόγους, έναν για κάθε άσκηση. Κάθε κατάλογος θα περιλαμβάνει τον πηγαίο, τον εκτελέσιμο κώδικα της άσκησης και αρχείο κειμένου (pdf). Φυσικά θα πρέπει να υπάρχει πληροφορία για τα ονόματα και τα ΑΜ των μελών της ομάδας.
- Μην υποβάλετε συμπιεσμένα αρχεία .rar

Εργαστηριακή Άσκηση 1 (Αναζήτηση σε λαβύρινθο) (20%)

Να κατασκευάσετε πρόγραμμα αναζήτησης για την επίλυση του ακόλουθου προβλήματος **πλοήγησης ρομπότ σε λαβύρινθο**:

Κατασκευή λαβύρινθου: Θεωρούμε ένα πλέγμα (πίνακα) με NxN κελιά, κάποια από τα οποία είναι ελεύθερα, ενώ τα υπόλοιπα περιέχουν εμπόδια και δεν μπορούμε να τα επισκεφθούμε. Κάθε κελί (x,y) χαρακτηρίζεται ως ελεύθερο ή όχι αποφασίζοντας ανεξάρτητα με πιθανότητα p. Τα N και p καθορίζονται στην αρχή του προγράμματος. Κάθε κελί έχει πλευρά μήκους 1.

Θέλουμε να βρούμε τη **βέλτιστη διαδρομή** (εάν υπάρχει) που πρέπει να ακολουθήσει το ρομπότ ξεκινώντας από ένα αρχικό κελί (S) ώστε να φτάσει σε κάποιο τελικό κελί (G). Οι συντεταγμένες (x,y) των κελιών S και G δίνονται από τον χρήστη στην αρχή του προγράμματος.

Μετακίνηση ρομπότ: i) Το ρομπότ μπορεί κάθε φορά να μετακινείται **είτε οριζόντια είτε κατακόρυφα είτε διαγώνια** σε ένα **γειτονικό ελεύθερο** κελί. **Όλες αυτές οι κινήσεις έχουν κόστος 1.**

ii) Επιπλέον το ρομπότ μπορεί να μεταφέρεται απευθείας (σε ένα βήμα) από το κάτω αριστερό κελί (έστω κελί Α) στο άνω δεξιό κελί (έστω κελί Β) ή αντίστροφα. Αυτή η απευθείας μεταφορά ανάμεσα στα κελιά Α και Β έχει κόστος 2. Φροντίστε κατά την κατασκευή του λαβύρινθου ώστε τα κελλιά Α και Β να είναι ελεύθερα.

Για το παραπάνω πρόβλημα να υλοποιήσετε:

i) αναζήτηση ομοιόμορφου κόστους (UCS)

ii) αναζήτηση Α* χρησιμοποιώντας όσο το δυνατόν καλύτερη **αποδεκτή ευρετική** συνάρτηση h(n). Θα πρέπει να εξηγήσετε γραπτώς (σε έγγραφο κειμένου pdf) γιατί η συνάρτηση h(n) που σκεφτήκατε είναι αποδεκτή.

Για κάθε λαβύρινθο που θα εξετάζετε, να εφαρμόζετε τόσο την μέθοδο UCS όσο και την μέθοδο Α* θεωρώντας την ίδια αρχική και τελική κατάσταση για να μπορείτε να συγκρίνετε τις μεθόδους.

Πιο συγκεκριμένα, για κάθε μέθοδο να τυπώνετε: α) τον λαβύρινθο, τα κελιά S, G και το μονοπάτι που βρήκατε, β) το κόστος του μονοπατιού αυτού, και γ) τον αριθμό των επεκτάσεων που έγιναν.

Να αναφέρετε στο κείμενο (report.pdf) τα αποτελέσματα καθώς και τα γενικά συμπεράσματά σας σχετικά με την αποδοτικότητα της Α* σε σχέση με τη UCS (αριθμός επεκτάσεων και εάν βρίσκουμε την ίδια λύση).

Σημείωση: για να βαθμολογηθεί η άσκηση είναι υποχρεωτικό να υλοποιήσετε και να εκτελέσετε και UCS και Α* και να υποβάλετε report.

Εργαστηριακή Ασκηση 2 (Κατασκευή Παιγνίου) (10%)
Να κατασκευάσετε πρόγραμμα (παίκτης MAX) το οποίο θα παίζει ενάντια σε κάποιο χρήστη (παίκτης MIN) το παρακάτω παίγνιο δύο παικτών που παίζουν εναλλάξ.

Θεωρούμε ένα **πλέγμα 4x3 και δύο γράμματα: Χ και Ο.**

Κάθε παίκτης, όταν έρθει η σειρά του, μπορεί να τοποθετήσει το γράμμα Χ (ο MAX) ή το γράμμα Ο (ο MIN) σε οποιαδήποτε κενή θέση του πλέγματος. **Υπάρχουν δύο τελικές τριάδες: 'XOX' και 'OXO'**. Αν με την τοποθέτησή ενός παίκτη σχηματίζεται κάποια από τις τελικές τριάδες είτε οριζόντια είτε κατακόρυφα είτε διαγώνια σε διαδοχικές θέσεις του πλέγματος, τότε το παίγνιο τερματίζει. Αν έχει σχηματιστεί η τριάδα **'XOX'** νικητής είναι ο MAX, αλλιώς (για την τριάδα **'OXO')** νικητής είναι ο MIN. Επίσης το παίγνιο τερματίζει όταν γεμίσει το πλέγμα χωρίς να έχει σχηματιστεί κάποια από τις επιθυμητές τριάδες (ισοπαλία).

Για να μειωθεί ο χώρος καταστάσεων, να ξεκινάτε το παίγνιο από αρχική κατάσταση όπου έχουν τοποθετηθεί ένα Χ και ένα Ο σε **μη διαδοχικές θέσεις** (οριζόντια, κατακόρυφα ή διαγώνια) του πλέγματος. Πρώτος παίζει ο MAX.

Αφού πρώτα ορίσετε κατάλληλες τιμές για την αξία των τελικών καταστάσεων, να κατασκευάσετε το πρόγραμμα εκτέλεσης του παιγνίου στο οποίο **ο MAX πρέπει να παίζει βέλτιστα εκτελώντας τον αλγόριθμο MINIMAX με ρίζα την τρέχουσα κατάσταση για να αποφασίσει για την κίνηση που θα κάνει κάθε φορά**. (Η υλοποίηση του MINIMAX να γίνει με τη χρήση αναδρομής. Να μην χρησιμοποιηθεί κλάδεμα α-β ή ευρετική συνάρτηση αξίας).

Να επισυνάψετε αρχείο κειμένου που να αναφέρει το πώς ορίζετε την κατάσταση του παιγνίου, πώς ορίζετε την αξία των τελικών καταστάσεων και να περιέχει μια σύντομη περιγραφή του κώδικα που αναπτύξατε.